MMCCMB1200 Controller and Memory Board (CMB1200) User's Manual (Revision 2)

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

The M•CORE name and logotype and the OnCE name are trademarks of Motorola, Inc.

© Motorola, Inc. 1998, 1999

CAUTION: ESD Protection

M•CORE development systems include open-construction printed circuit boards that contain static-sensitive components. These boards are subject to damage from electrostatic discharge (ESD). To prevent such damage, you must use static-safe work surfaces and grounding straps, as defined in ANSI/EOS/ESD S6.1 and ANSI/EOS/ESD S4.1. All handling of these boards must be in accordance with ANSI/EAI 625.

Table of Contents

Section 1 Introduction

1.1 CMB1200 Features 9
1.2 System and User Requirements 10
1.3 CMB1200 Layout 10
1.4 Foot Placement 12

Section 2 Configuration

2.1 Setting Jumper Headers 13
2.1.1 Setting the SRAM/FLASH Chip Select Header (W1) 15
2.1.2 Setting the FLASH Byte/Word Select Headers (W3, W6) 16
2.1.3 Setting the Startup Vector Select Header (W9) 16
2.1.4 Setting the Serial Port Enable Header (W14) 17
2.1.5 Setting the Core Voltage Header (W19) 18
2.2 Making Computer-System Connections 18
2.3 Performing the CMB1200 Self-Test 19
2.4 Memory Maps 20

Section 3 Operation

3.1 Debugging Embedded Code 21
3.1.1 Using the Picobug Debug Monitor 21
3.1.2 Picobug Sample Session 23
3.1.3 Using the GNU Source-Level Debugger 25
3.2 Downloading to FLASH Memory 26
3.2.1 Using the FLASH Programmer 26
3.2.2 Restoring System Software 31
3.3 Controlling CMB1200 LEDs 31

Section 4 Connector Information

4.1 MAPI Connectors (P1/J1, P2/J2, P3/J3, P4/J4) 33
4.2 OnCE Connector (P5) 43

Index

Table of Contents

List of Figures

Figure 1-1 MMCCMB1200 Computer and Memory Board 11
Figure 1-2 Foot Locations (on Bottom of CMB1200). 12
Figure 3-1 FLASH Programmer Main Screen 26
Figure 3-2 Picobug Protocol Selection 27
Figure 3-3 Memory Contents Display 30
Figure 4-1 MAPI Connector P1/J1 Pin Assignments. 34
Figure 4-2 MAPI Connector P2/J2 Pin Assignments. 36
Figure 4-3 MAPI Connector P3/J3 Pin Assignments. 38
Figure 4-4 MAPI Connector P4/J4 Pin Assignments. 41
Figure 4-5 OnCE Connector P5 Pin Assignments. 43

List of Figures

List of Tables

Table 1-1 MMCCMB1200 Controller and Memory Board Specifications . . 11
Table 2-1 Jumper Header Positions 13
Table 2-2 W14 Jumper Signals 17
Table 2-3 MMC2001 Address-Decoded Memory Map 20
Table 3-1 Picobug Commands 22
Table 3-2 LED Registers and Addresses 31
Table 4-1 MAPI Connector P1/J1 Signal Descriptions 35
Table 4-2 MAPI Connector P2/J2 Signal Descriptions 37
Table 4-3 MAPI Connector P3/J3 Signal Descriptions 39
Table 4-4 MAPI Connector P4/J4 Signal Descriptions 42
Table 4-5 OnCE Connector P5 Signal Descriptions 43

List of Tables

Section 1 Introduction

This user's manual explains connection, configuration, and operation information for the MMCCMB1200 Computer and Memory Board (CMB1200), a development tool of Motorola's M•CORE™ family.

As a standalone tool, the CMB1200 lets you use the Picobug debug monitor, through an RS232 connection to your computer. In conjunction with the Picobug monitor, you may use the GNU source-level debugger. The CMB1200 also has a OnCE™ connector, enabling you to use a debugging application that requires one.

Motorola's FLASH programmer lets you download your code into the CMB1200's SRAM (for execution) or FLASH memory (for execution or for storage in non-volatile memory). Should your application overwrite factory programming in the FLASH memory device, you can use the FLASH programmer to restore the factory programming.

The CMB1200 will combine easily with other, optional development boards from Motorola. Such optional boards will expand CMB1200 capacity, enhance CMB1200 performance, or add to CMB1200 features.

1.1 CMB1200 Features

The CMB1200 features:

- 512K-by-16 FLASH memory, at 90 nanoseconds
- Two 256K-by-16 SRAM memories, at 12 nanoseconds
- Two power regulators: 3.3 volts and 2.0 volts.
- Power supply that converts line power to 5-volt power.
- Two RS232 channels for serial communications. These channels use internal universal asynchronous receiver/transmitters (UARTs).
- A OnCE connector
- A MAPI 400 connector interface ring, on the top and bottom of the CMB1200, for easy connection to other, compatible development boards.
- Address decoding via a 16V8 generic array logic (GAL) device.
- Jumper headers for such configuration items as power select, SRAM and address decode, and byte/word FLASH configuration.
- Motorola's FLASH programmer.
- Picobug debug monitor and GNU source-level debugger (from the Free Software Foundation).

- Ability to use debugging software from one or more other manufacturers. (An example is SingleStep debugging software, from Software Development Systems.)

1.2 System and User Requirements

You need an IBM PC or compatible computer, running the Windows 95 or WindowsNT (version 4.0) operating system. The computer requires a Pentium (or equivalent) microprocessor, 16 megabytes of RAM, 50 megabytes of free hard-disk space, an SVGA color monitor, and an RS232 serial-communications port. To use the Picobug debug monitor, you also need Hyperterminal or a comparable terminal-emulation program.

To get the most from your CMB1200, you should be an experienced C or M•CORE assembly programmer.

Your CMB1200 requires 5-volt (± 0.25 volt) input power, at 250 milliamperes. The power supply that comes with your CMB1200 provides 5 volts from line power.

1.3 CMB1200 Layout

Figure 1-1 shows the layout of the CMB1200. Jumper header W1 configures the SRAM and FLASH-memory chip selects, which use generic array logic (GAL) address decoding. Jumper headers W3 and W6 configure byte or word mode for FLASH memory. Jumper header W9 selects the startup vector. Jumper header W14 enables or disables the serial ports, and also configures the CMB1200 for its self-test. Jumper header W19 applies 3.3 or 2.0 volts to the core power plane.

Connectors P1 through P4, on the top of the board, are the MAPI I/O and interrupt connectors (the corresponding MAPI connectors on the bottom of the CMB1200 are J1 through J4). Connector P5 is the OnCE connector. Connector J8 is the power connector. Connectors J9 and J10 are the RS232 serial communication connectors.

Switch S1 is the reset switch. Location F1 is for the CMB1200 fuse.

Yellow LEDs ST0 through ST3 light to confirm that the CMB1200 passed its self-test. (User programs also can control these LEDs.) Green LED DS5 lights to confirm that the CMB1200 is receiving +5-volt input power.

The MMC2001 resident MCU is at location U7. SRAM memory devices are at locations U1 and U3. The FLASH memory device is at location U2.

.

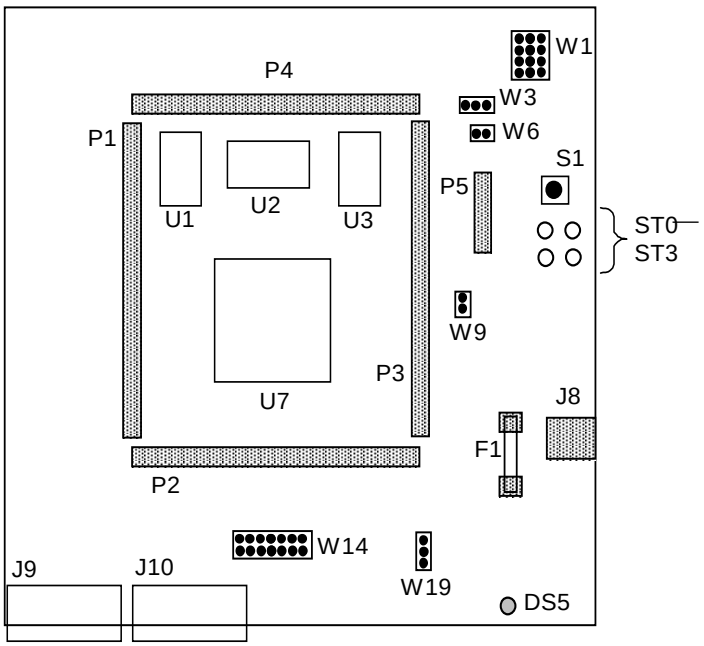


Figure 1-1 MMCCMB1200 Computer and Memory Board

Table 1-1 lists CMB1200 specifications.

Table 1-1 MMCCMB1200 Controller and Memory Board Specifications

Characteristic	Specifications
MCU extension I/O ports	HCMOS compatible
Operating temperature	0° to 40° C
Storage temperature	-40° to +85° C
Relative humidity	0 to 90% (non-condensing)
Clock	32 MHz
Power requirements	5 volts dc, at 250 milliamperes, provided from a separate power source
Dimensions	5.75 x 6.0 inches (146 x 152 mm)
Weight	5.0 ounces (142 g)

1.4 Foot Placement

Your CMB1200 comes from the factory ready for use with other, compatible development boards.

For standalone use, however, you should attach the four rubber feet to the bottom of the CMB1200. **Figure 1-2** shows the correct locations.

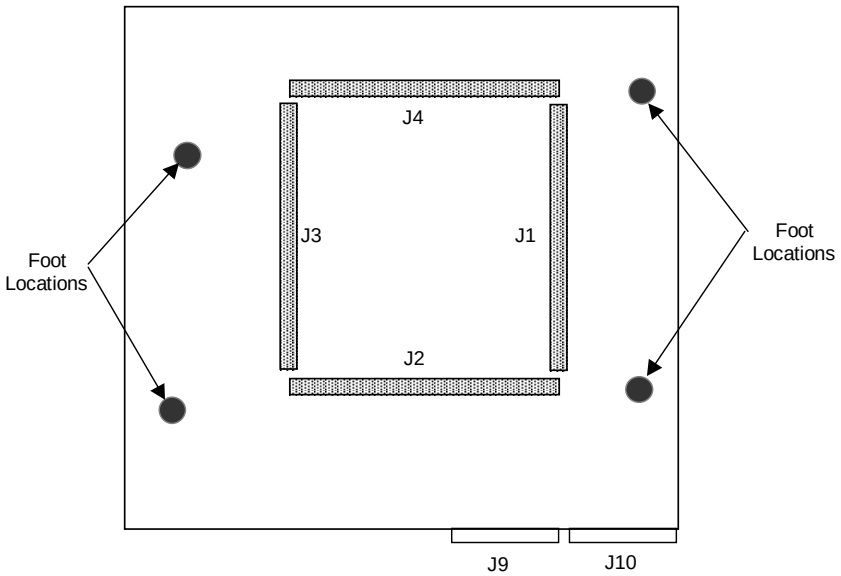


Figure 1-2 Foot Locations (on Bottom of CMB1200)

NOTE: *The correct locations for the rubber feet are not the extreme corners of the CMB1200. The correct locations do not form a symmetrical pattern.*

Section 2 Configuration

This chapter explains how to configure your CMB1200, and how to hook it up to your computer system.

2.1 Setting Jumper Headers

Your CMB1200 has six jumper headers. **Table 2-1** is a summary of settings for these headers; **2.1.1 Setting the SRAM/FLASH Chip Select Header (W1)** through **2.1.5 Setting the Core Voltage Header (W19)** give additional information.

NOTE: *CMB1200 locations W2, W4, W5, W7, W8, W10, W11, W12, W13, W15, W16, W17, and W18 are not populated with jumper headers.*

Table 2-1 Jumper Header Positions

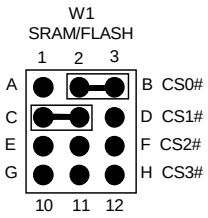
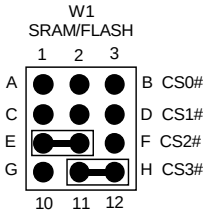
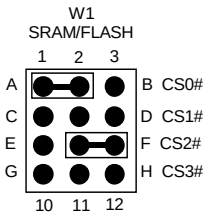
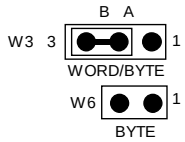
Header	Position	Effect
SRAM/FLASH Chip Select, W1 **(Only 2 jumpers in this header.)**		Uses the U6 GAL address-decode circuit: configures chip select 1 to control the SRAM and chip select 0 to control the FLASH. **Factory setting.**
		Uses the U6 GAL address-decode circuit: configures chip select 2 to control the SRAM and chip select 3 to control the FLASH. (Another of many possible configurations.)
		Uses the U6 GAL address-decode circuit: configures chip select 0 to control the SRAM and chip select 2 to control the FLASH. (Another of many possible configurations.)
FLASH Byte/Word Select, W3, W6		Configures word mode for FLASH. **Factory setting.**

Table 2-1 Jumper Header Positions (Continued)

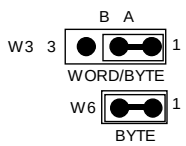
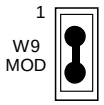
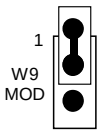
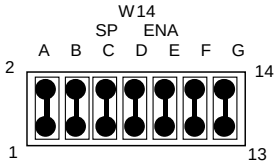
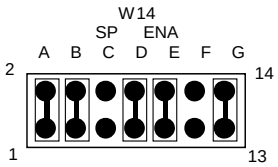
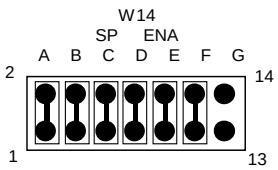
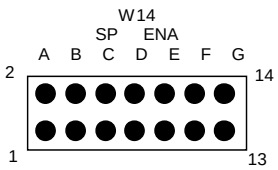
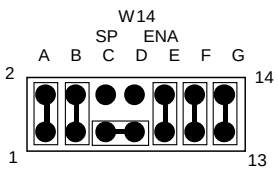
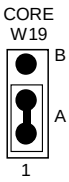
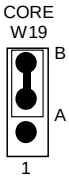
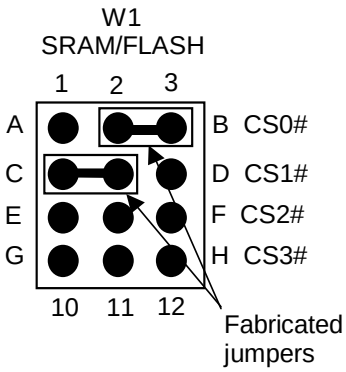
Header	Position	Effect
FLASH Byte/Word Select, W3, W6 (continued)		Configures byte mode for FLASH.
Startup Vector Select, W9		Upon power-up, selects the startup vector from the lowest address of the external chip-select 0 device. (Usually this is FLASH memory.) Correct configuration for CMB1200 self-test. **Factory setting.**
		Upon power-up, selects the startup vector from the lowest address of MCU internal ROM. Invokes the Mbug debugger. (Also the correct setting for the FLASH programmer.)
Serial Port Enable, W14		Connects all serial-port (UART) signals to RS232 transceivers. **Factory setting.**
		Connects port 0 serial-port signals, disconnects port 1 serial port signals.
		Disables RS232 drivers and receivers, but maintains UART-signal connections to RS232 transceivers.
		Disconnects all serial-port signals.
		Configures the CMB1200 self-test.

Table 2-1 Jumper Header Positions (Continued)

Header	Position	Effect
Core Voltage, W19		Applies 3 volts to the core power plane. Factory setting.
		Applies 2-volts to the core power plane.

2.1.1 Setting the SRAM/FLASH Chip Select Header (W1)

Jumper header W1 selects chip selects, using the generic array logic (GAL) address decoder. The diagram below shows the factory configuration: the fabricated jumper at position B specifies that chip select 0 controls the FLASH memory; the fabricated jumper at position C specifies that chip select 1 controls the SRAM.



Alternatively, you can select any other chip select to control the SRAM, then select any of the remaining three chip selects to control the FLASH.

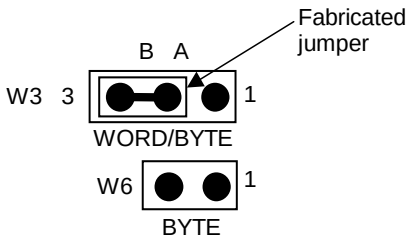
For example, to have chip select 2 control the SRAM, put the fabricated jumper at W1 position E. Then select chip select 0, 1, or 3 to control the FLASH by putting the second fabricated jumper at position B, D, or H, respectively.

(**2.4 Memory Maps** explains the memory map address range for each chip select.)

NOTE: *Use only two jumpers in header W1: one in position A, C, E, or G, and the second in one of the three remaining positions of B, D, F, or H.*

2.1.2 Setting the FLASH Byte/Word Select Headers (W3, W6)

Jumper headers W3 and W6 determine whether the CMB1200 uses FLASH in word or byte mode. The diagram below shows the factory configuration: the fabricated jumper at W3 position B and no fabricated jumper in header W6 configures word mode.



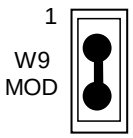
To configure byte mode for the FLASH:

- Reposition the W3 jumper to position A, and
- Put a jumper in header W6.

NOTE: *Do not put the W3 jumper in position A unless you also put a jumper in header W6. Do not put a jumper in header W6 if you put the W3 jumper in position B.*

2.1.3 Setting the Startup Vector Select Header (W9)

Jumper header W9 selects the startup vector that the MCU uses upon power-up. The diagram below shows the factory configuration: the fabricated jumper installed in this header selects the vector in the external chip-select 0 device (usually the FLASH memory).



Alternatively, you can select the vector in MCU internal ROM. To do so, remove the jumper from header W9.

2.1.4 Setting the Serial Port Enable Header (W14)

Jumper header W14 connects or disconnects serial-port (UART) signals from RS232 transceivers. The diagram below shows the factory configuration: the seven fabricated jumpers, in positions A through G, connect the UART signals.

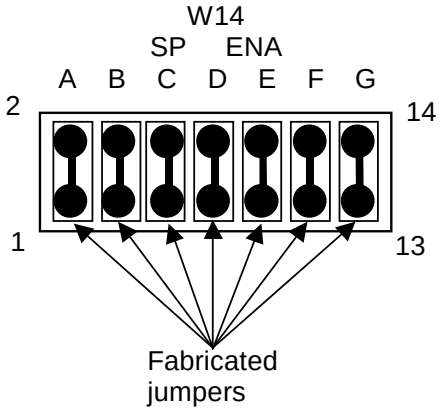


Table 2-2 lists the serial-port signals that correspond to each W14 jumper position. To disconnect a signal, making it available for other use, remove the corresponding jumper. (**Table 2-1** shows such a possible configuration: jumpers removed from positions C and F, disconnecting the port 0 signals.)

Table 2-2 W14 Jumper Signals

UART Signal	Jumper Position
TXD0 – (Transmit data, port 0)	A
CTS0 – (Clear to send, port 0)	B
TXD1 – (Transmit data, port 1)	C
RXD0 – (Receive data, port 0)	D
RTS0 – (Ready to send, port 0)	E
RXD1 – (Receive data, port 1)	F
Transceiver Standby[1]	G

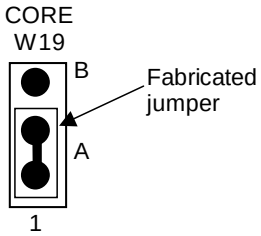
NOTES:

1. Removing the jumper from position G disables RS232 driver and receiver outputs by putting them in a high-impedance state. (**Table 2-1** shows an example of such a configuration.) To make sure that UART signals of other positions are connected to active RS232 transceivers, make sure that a jumper is in position G.

NOTE: *To configure a CMB1200 self-test, remove the jumpers from W14 positions C and D, then insert one of the jumpers turned 90 degrees, so that the jumper connects W14 pins 5 and 7. At the end of the self-test, restore W14 to its correct configuration for operation.*

2.1.5 Setting the Core Voltage Header (W19)

Jumper header W19 applies 3.3 or 2.0 volts to the core power plane. The diagram below shows the factory configuration: the fabricated jumper installed in position A applies 3.3 volts to the core power plane.



Alternatively, you can apply 2 volts to the power plane. To do so, move the jumper to W19 position B.

NOTE: *Removing the jumper from header W19 removes power from the power plane, so usually is not appropriate. An exception would be temporarily replacing the jumper with an ammeter, to measure the current.*

2.2 Making Computer-System Connections

When you have configured your CMB1200, you are ready to connect it to your computer system:

1. If you will use the Picobug-GNU debug monitor software, the Motorola FLASH programmer, or another application that needs RS232 connection to port 0, connect an RS232 cable between CMB1200 connector J10 and the appropriate serial port of your computer. If your application will use port 1, connect the port 1 RS232 cable between CMB1200 connector J9 and the appropriate serial port of your computer.
2. If your application will use the OnCE connector, connect a OnCE-compatible cable between CMB1200 connector P5 and the OnCE interface device of your computer. (**Section 4 Connector Information** gives pin assignments and signal descriptions for connector P5.)
3. Make sure that your +5-volt power supply is turned off (or disconnected from line power). Connect the power supply's ground wire to pin 1 (black) of CMB1200 connector J8. Connect the power supply's power wire to pin 2 (red) of connector J8.
4. If you have not already done so, apply power to your computer. Turn on your +5-volt power supply (or connect it to line power): the green LED (DS5) lights to confirm that the EVB is powered. (Should the DS5 LED *not* light, you may need to replace the fuse at location F1, next to power connector J8. Use a BUS GMA-1.5A fuse, or compatible.)

5. This completes system connections; you are ready to perform a CMB1200 self-test, per the instructions of **2.3 Performing the CMB1200 Self-Test**, below. You are ready to begin debugging or other development activities, per the instructions of **Section 3 Operation**.
6. (Optional) For scope observation of specific I/O or interrupt signals, you may use connectors P1 through P4. You may clip individual lines to the pins of these connectors, or use appropriate connectors. (**Section 4 Connector Information** gives pin assignments and signal descriptions for these connectors.)

2.3 Performing the CMB1200 Self-Test

Once you have configured your CMB1200, you can perform a self-test of CMB1200 components.

1. If your CMB1200 is powered, turn off or disconnect power. The green LED (DS5) goes out.
2. Make sure that a fabricated jumper is in jumper header W9.
3. Remove the fabricated jumpers from positions C and D of jumper header W14. Turn one of these jumpers 90 degrees, then reinsert it, so that it connects W14 pins 5 and 7.
4. Apply power to the CMB1200. The green LED (DS5) comes on to confirm power. The CMB1200 automatically begins its self-test.
5. The four yellow LEDs (ST0 through ST3) flicker during the self-test (approximately one minute), then light without flickering to confirm that the CMB1200 passed the self-test.
6. If, at the end of the self-test, any of the yellow LEDs does not light without flickering, the CMB1200 fails the self-test. Contact Motorola customer support for assistance.
7. When the CMB1200 passes the self-test, disconnect power, restore jumper header W14 to its operating configuration, then restore power. This completes the self-test.

2.4 Memory Maps

Your CMB1200 uses a 16V8 generic array logic (GAL) device for chip-select address decoding. The configuration of jumper header W1 determines which chip selects are active. **Table 2-3** lists the memory-map starting and ending addresses for all such possibilities (bold type indicates the factory configuration).

Table 2-3 MMC2001 Address-Decoded Memory Map

Chip Select	Starting Address	Ending Address	W1 Position	Memory Resource
CS0	0x2D00_0000	0x2D0F_FFFF	A	SRAM
	0x2D00_0000	**0x2D0F_FFFF**	**B**	**FLASH**
CS1	**0x2F00_0000**	**0x2F0F_FFFF**	**C**	**SRAM**
	0x2F00_0000	0x2F0F_FFFF	D	FLASH
CS2	0x2E00_0000	0x2E0F_FFFF	E	SRAM
	0x2E00_0000	0x2E0F_FFFF	F	FLASH
CS3	0x2C00_0000	0x2C0F_FFFF	G	SRAM
	0x2C00_0000	0x2C0F_FFFF	H	FLASH

As the table shows, the factory configuration for jumper header W1:

- Puts FLASH memory into the chip-select 0 memory space (addresses 0x2D00_0000 through 0x2D0F_FFFF), and
- Puts SRAM memory into the chip-select 1 memory space (addresses 0x2F00_0000 through 0x2F0F_FFFF).

Section 3 Operation

This chapter explains how to begin using debugging tools available for your MMCCMB1200 Controller and Memory Board, as well as how to use Motorola's FLASH programmer.

3.1 Debugging Embedded Code

With your CMB1200, you may use the Picobug debug monitor, as standalone software. Optionally, you may use the GNU source-level debugger with the Picobug monitor. Other firms may produce additional software to run, test, and modify the code you develop for embedding in an MMC2001 MCU.

3.1.1 Using the Picobug Debug Monitor

The Picobug debug monitor comes burned into the FLASH memory device of your CMB1200 (location U3). Before you start the Picobug monitor, make sure that you have an RS232 connection between CMB1200 connector J10 and a serial port of your computer.

To start the Picobug monitor, for use as a standalone debugger:

1. Make sure that power is *not* applied to your CMB1200.
2. Activate Hyperterminal or a comparable terminal-emulation program. (If you use a different terminal-emulation program, you must make corresponding changes in the commands and menu selections of these instructions, and in the instructions of **3.1.2 Picobug Sample Session**.)
3. From the File menu, select Properties. This opens a properties dialog box.
4. Click on the Configure button of the dialog box. This opens a configuration dialog box.
5. Use the configuration dialog box to set the operating speed (19,200) and the correct communications port (for example, COM1). Click the OK button of the dialog box.
6. Apply power to the CMB1200 and press the enter key. The Picobug monitor starts automatically, displaying the command prompt: `picobug>`.

To use the Picobug debug monitor, merely enter commands at the prompt. **Table 3-1** explains these commands. To see a list of these commands on your computer screen, enter a question mark or the extra command `help` at the command prompt.

Table 3-1 Picobug Commands

Command	Explanation
br [*address*]	Breakpoint: • With optional *address* value, sets a new breakpoint at that address. • Without any *address* value, lists all current breakpoints.
g [*address*]	Go: • With optional *address* value, starts code execution from that address. • Without any *address* value, starts code execution from the current program-counter value. In either case, execution stops when it arrives at a breakpoint.
gr	Go to Return: Executes code from the current program-counter value to the return address of the calling routine. (Should execution arrive at a breakpoint before encountering the return address, execution stops at the breakpoint.)
gt *address*	Go to Address: Executes code from the current program-counter value to the specified *address* value. (Should execution arrive at a breakpoint before encountering the specified address, execution stops at the breakpoint.)
lo [*address*]	Download: • With optional *address* value, downloads a binary image to that address in SRAM. • Without any *address* value, downloads to SRAM an S-record text file.
md *address1 address2* [;*size*]	Memory Display: Displays memory contents from the *address1* value to the *address2* value. The optional *size* value specifies the format: **b** (bytes, the default), **h** (half words), **w** (words), or **i** (instructions).
mds *address* [;*size*]	Memory Display: Displays 256 bytes of memory contents, beginning at the *address* value. The optional *size* value specifies the format: **b** (bytes, the default), **h** (half words), **w** (words), or **i** (instructions).
mm *address* [value] [;*size*]	Modify Memory: • With optional *value* parameter value, assigns that value to the *address* location. • Without any *value* parameter value, prompts for a value for the *address* location, then prompts for a new value for the next location. To stop modification, enter a period instead of a new value. • The optional *size* value, specifies the format: **b** (bytes, the default), **h** (half words), **w** (words), or **i** (instructions).
nobr [*address*]	No Breakpoint: • With optional *address* value, removes the breakpoint from that address. • Without any *address* value, removes all the breakpoints.
reset	Reset: Resets the CPU and peripherals.
rd [*name*]	Register Display: • With optional *name* value, displays the value of that CPU register. • Without any *name* value, displays the values of all CPU registers.
rm *name value*	Register Modify: Assigns the *value* parameter value to the *name* CPU register.
t	Trace (Step): Single steps one instruction; identical to the `s` command.
s	Step (Trace): Single steps one instruction; identical to the `t` command.

3.1.2 Picobug Sample Session

1. This sample session begins with the Picobug prompt:

```
picobug
```

2. To see the contents of all registers, enter the Register Display (rd) command without any name value:

```
picobug>    rd
```

   The system responds with a display such as this:

```
     pc 30000286  epc      30000286  fpc      0010a000
    psr 80000100  epsr     80000100  fpsr     00020000
ss0-ss4 bad0beef  20000c00 20008000  20010042 00000801  vbr       30005c00
  r0-r7 300027f8  00000050 0000ea60  30005f94 30006708  80070101  00000200 00000040
 r8-r15 0010a000  00020000 20000c00  300067c0 00000000  30005f94  10005000 30000286
```

3. To see the contents of a specific register, such as the epc register, enter the Register Display (rd) command *with* the name value:

```
picobug> rd epc
```

   The system responds with a display such as this:

```
epc:     30000286
```

4. To see the contents of a specific memory location, enter the Memory Display (md) command with the location address. An optional size value (in this case w, for word) may be part of the command:

```
picobug> md    0x30001000  ;w
```

   The system responds with a display such as this:

```
30001000:  8EF0B37E
```

5. To see the contents of a memory range, enter the Memory Display (md) command with the beginning and ending addresses. An optional size value (in this case b, for byte) may be part of the command:

```
picobug> md    0x30000000  0x30000010  ;b
```

   The system responds with a display such as this:

```
30000000: 8E  F0  00  00  55  55  55  55  0E  22  9E  E8  03  20  0D  20  0E $..UUUU. "… . .
30000010: F7
```

6. To download into SRAM a program executable, in S-record format, enter the Download (lo) command without any address value:

```
picobug> lo
```

The system waits for you to send the program executable file. To do so, open the Transfer menu and select Send Text File. This opens a file-select dialog box. Use this dialog box to specify the appropriate S-record file, then click on the Open button. When the file-select dialog box disappears, press the Enter key of your keyboard. (Pressing the keyboard Enter key adds a necessary line-feed character to the end of the S-record file.) As soon as the download is complete (this may take several minutes), the Picobug prompt reappears:

```
picobug>
```

7. To see the new contents of registers, enter the Register Display (rd) command again, without any name value:

```
picobug> rd
```

The system responds with an updated display, which shows that the pc register value reflects the start of the program just downloaded:

```
     pc 3000022a  epc      2d00108a  fpc      0010a000
    psr 80000000  epsr     80070101  fpsr     00020000
ss0-ss4 bad0beef  20000c00 20008000  20010042 00000801  vbr       30005c00
  r0-r7 bad0beef  00000050 00000000  d89f69ab 30005f20  80000000  00000200 00000024
 r8-r15 0010a000  300066b8 300067d7  30006948 30006714   30006708  300067c8 2d0001c4
```

8. To set a breakpoint at address 0x3000025C, enter this address as part of the Breakpoint (br) command:

```
 picobug> br   0x3000025c
```

The Picobug prompt reappears, confirming that the system set the breakpoint:

```
picobug>
```

9. To see the list of breakpoints, enter the Breakpoint (br) command *without* any address value:

```
picobug> br
```

The system responds with the addresses of breakpoints, in this case only the breakpoint set in step 8:

```
3000025C
```

10. To start program execution, enter the Go (g) command:

```
picobug> g
```

In this instance, the breakpoint set during step 8 stops code execution. The system responds with this new display of register values:

```
 At breakpoint!!
    pc 3000025c  epc      3000025c  fpc      0010a000
   psr 80000100  epsr     80000100  fpsr     00020000
ss0-ss4 bad0beef  20000c00 20008000  20010042 00000801  vbr       30005c00
  r0-r7 300027f8  00000050 0000ea60  d89f69ab  30005f20 80000000  00000200 00000040
 r8-r15 0010a000  300066b8 300067d7  30006948 30006714  30006708  10005010 300002a2
```

11. To remove all breakpoints, enter the No Breakpoint (nobr) command, without any address value:

```
picobug> nobr
```

The Picobug prompt reappears, confirming that the system has removed the breakpoints:

```
picobug>
```

12. To see the list of breakpoints again, once more enter the Breakpoint (br) command without any address value:

```
picobug> br
```

As there are no longer any breakpoints, the system responds with the Picobug prompt:

```
picobug>
```

13. To continue with this example session, enter another appropriate command. For example, to resume program execution, enter the Go (g) command.

14. To end your Picobug session, remove power from the EVB and close the terminal-emulation program.

3.1.3 Using the GNU Source-Level Debugger

The GNU source-level debugger is on the CD-ROM that comes with your CMB1200. This GNU software works with the Picobug debug monitor to provide source-level debugging for your code.

The CMB1200 software release guide gives the instructions for loading the GNU software, and for making any connections different from standalone Picobug connections.

3.2 Downloading to FLASH Memory

Motorola's FLASH programmer lets you program your code into FLASH memory, verify that FLASH contents match those of a download file, display the contents of FLASH memory, erase FLASH memory, or erase a sector of FLASH memory.

3.2.1 Using the FLASH Programmer

Follow these steps to use the programmer:

1. If you have not already installed the FLASH programmer onto your computer hard disk, do so. The CMB1200 product release guide includes installation instructions.
2. Reset the CMB1200, by pressing switch S1.
3. Start the FLASH programmer. The main screen (**Figure 3-1**) appears.

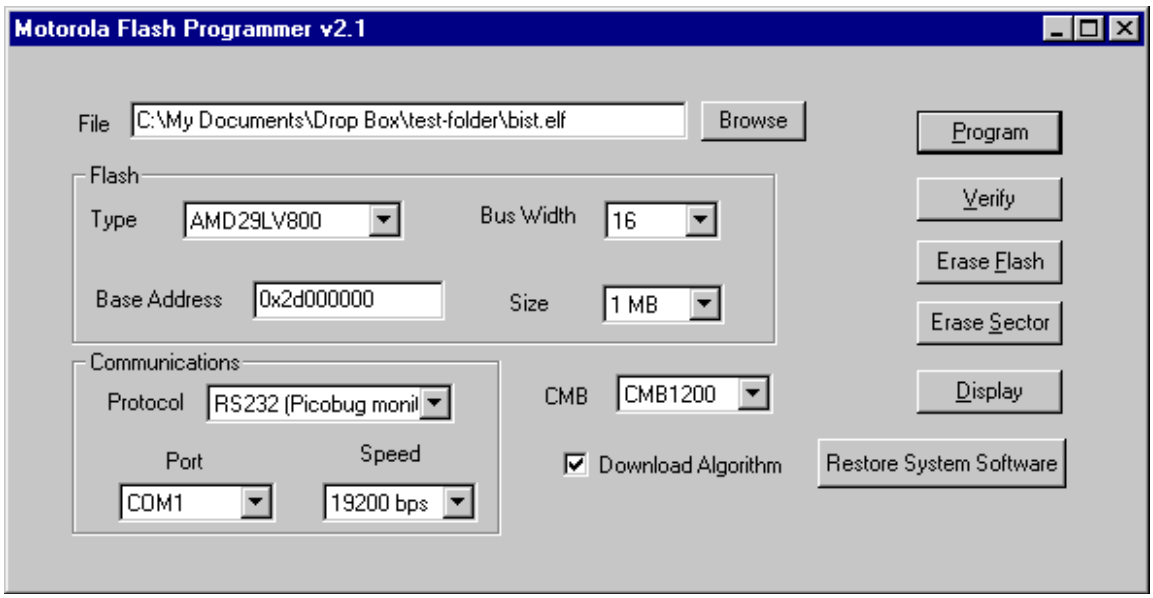


Figure 3-1 FLASH Programmer Main Screen

NOTE: ***Figure 3-1*** *shows field values appropriate for using the FLASH programmer with a CMB1200 in standalone mode. Except as explained in the instructions below, selecting different values for these fields could cause errors.*

4. Go to the File field, at the upper left of the main screen.
   - If you know the full pathname of the file to be programmed, enter the pathname in this field.

- If you do not know the full pathname of the file to be programmed, click on the Browse button. This brings up a standard file-select dialog box: select the file and click on the OK button. This returns you to the main screen, entering the pathname in the File field.

5. Use the Flash area of the main screen to configure the FLASH type, bus width, and size.

   In the Base Address field, enter the FLASH starting address for the chip select you configured via jumper header W1:

   - For chip select 0, enter base address 0x2d000000.
   - For chip select 1, enter base address 0x2f000000.
   - For chip select 2, enter base address 0x2e000000.
   - For chip select 3, enter base address 0x2c000000.

6. In the Communications area of the main screen, use the Port field to specify the PC serial port, and use the Speed field to specify the communications rate (19200 bps).

   Use the Protocol field to specify the communications protocol:

   - If your computer connects to the CMB1200 through a serial port, and the CMB1200 system software is intact, specify RS232 (Picobug monitor). **Figure 3-2** illustrates this selection.

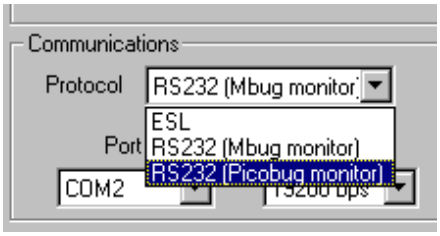


Figure 3-2 Picobug Protocol Selection

   - If your computer connects to the CMB1200 OnCE connector through an EBDI box, specify ESL.
   - If your computer connects to the CMB1200 through a serial port, but the CMB1200 system software has been deleted, specify RS232 (Mbug monitor).

NOTE: *If you select the RS232 (Mbug monitor) protocol, a prompt tells you to remove the W9 jumper and reset the CMB1200. Even if you just have reset the CMB1200, you must reset it again, after removing the W9 jumper.*

You may set the protocol, port, and speed only once during each programmer session. To make another change in any Communications-area setting, you must exit, then restart, the FLASH programmer.

7. Go to the CMB1200 field, near the lower center of the main screen. Make sure that the field specifies CMB1200. Make sure that the Download Algorithm box is checked.

   This completes the programmer setup. You are ready for the programming actions that correspond to the buttons along the right edge of the main screen.

8. To program FLASH memory, click on the Program button. The software downloads the file you specified, displaying a progress message. A Download successful message appears at the end of downloading: you are ready to use the code in FLASH memory.

   - If this is the first programming action of this FLASH programmer session, the software downloads an algorithm file before downloading the file you specified. A progress message appears during the downloading of this algorithm file.
   - If the software cannot find the algorithm file, an appropriate error message identifies the file. Click on the message's OK button to bring up a file-select dialog box, then use this dialog box to specify the location of the algorithm file. If necessary, recopy the file from the transmittal CD-ROM. Click on the OK button to resume programming FLASH memory.
   - The error message Unable to Validate Flash configuration indicates some problem with the programming. A likely such problem is that the chip select base address does not correspond to the configured chip select. Correct the problem, then click again on the Program button.

NOTE: *Another possible cause of validation failure is that the link control file of user code specifies SRAM, not FLASH, as the download destination. You must correct such a problem in the user code.*

9. To verify that the contents of Flash memory match the selected download file, click on the Verify button. A progress message appears as verification begins. A Verify successful message appears at the end of verification.

   - If this is the first programming action of this FLASH programmer session, the software downloads an algorithm file before verifying FLASH. A progress message appears during the downloading of this algorithm file. (Should the software be unable to find the algorithm file, an appropriate error message appears, as explained under the program FLASH memory step, above.)
   - If verification fails, an error message specifies the location that did not have the expected contents.
   - To recover from a verification failure, try programming Flash again, to replace the selected download file.

NOTE: *Another possible cause of verification failure is that the link control file of user code specifies SRAM, not FLASH, as the download destination. You must correct such a problem in the user code.*

10. To erase FLASH memory, click on the Erase Flash button. The programmer erases all contents of the FLASH memory except for sector 0 (which contains the system software). Erasing takes 20 to 30 seconds.

    If this is the first programming action of this FLASH programmer session, the software downloads an algorithm file before erasing FLASH. A progress message appears during the downloading of this algorithm file. (Should the software be unable to find the algorithm file, an appropriate error message appears, as explained under the program FLASH memory step, above.)

11. To erase a sector of FLASH memory, click on the Erase Sector button. This brings up the Flash Sector Number dialog box. Enter the number of the sector to be erased, then click on the OK button.

    - If this is the first programming action of this FLASH programmer session, the software downloads an algorithm file before erasing the FLASH sector. A progress message appears during the downloading of this algorithm file. (Should the software be unable to find the algorithm file, an appropriate error message appears, as explained under the program FLASH memory step, above.)
    - If you specify sector 0, a message reminds you that the system software is in this section. Buttons of the message box let you cancel the erasure or proceed with the erasure.

NOTE: *The target FLASH has 16 sectors, 0 through 15, each of 64 kilobytes. Do not erase sector 0, which contains system software, unless it is absolutely necessary. (If you must erase sector 0, you subsequently can restore factory programming by clicking on the Restore System Software button.* ***3.2.2 Restoring System Software*** *gives additional information about restoring sector 0.)*

12. To view the contents of Flash memory, click on the Display button. This brings up the Memory Contents display (**Figure 3-3**).

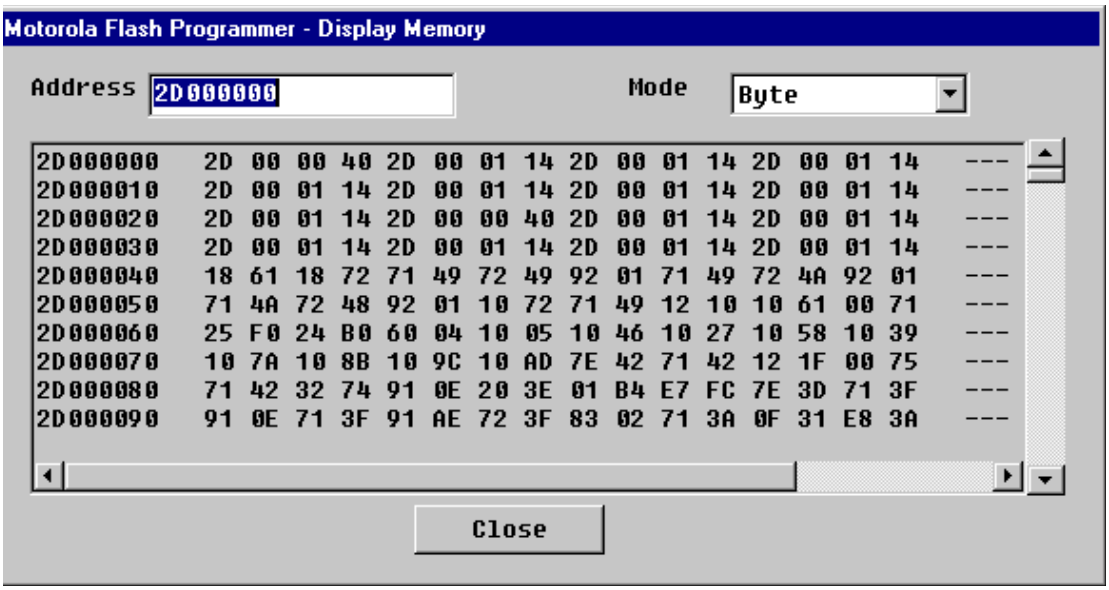


Figure 3-3 Memory Contents Display

- If this is the first programming action of this FLASH programmer session, the software downloads an algorithm file before displaying FLASH contents. A progress message appears during the downloading of this algorithm file. (Should the software be unable to find the algorithm file, an appropriate error message appears, as explained under the program FLASH memory step, above.)
- The Address field shows the first address of the value display. One way to change the display is to enter a different address in this field.
- Another way to change the value display is to use the vertical or horizontal scroll bars.
- Use the Mode field to specify byte, half-word, or word values in the display.
- When you are done viewing the display, click on the Close button to return to the main screen.

13. At the end of your programming session, click on the Exit button.

3.2.2 Restoring System Software

If you must overwrite FLASH-device sector 0, you subsequently may use the FLASH programmer to restore Picobug and other factory software to sector 0. To do so, select the RS232 (Mbug monitor) communications protocol. Then click on the Restore System Software button.

- If the system software is in your current hard-disk directory, the FLASH programmer automatically restores factory programming to FLASH sector 0. The main screen reappears to confirm successful programming.
- If you receive a message that the system software does not exist, it may be because the software is in a different hard-disk directory. If so, make that directory the active one and click again on the Restore System Software button.
- The CD-ROM that comes with your CMB1200 is yet another source of system software: files `Dev_Sys/MMC2001/SYSsw_FLASHprog/monitor.elf` and `Dev_Sys/MMC2001/SYSsw_FLASHprog/bist.elf`.

3.3 Controlling CMB1200 LEDs

Section 2 Configuration explained how the four yellow CMB1200 LEDs flicker and light as part of the CMB1200 self-test. Your own code also can control these LEDs, which are connected to pins PWM0 through PWM3 of the microprocessor.

Table 3-2 lists the control register names and addresses for each LED.

Table 3-2 LED Registers and Addresses

LED Location	Register	Address
ST0	PWM0	0x10005000
ST1	PWM1	0x10005008
ST2	PWM2	0x10005010
ST3	PWM3	0x10005018

To control an LED, your code must assign appropriate values to the LED's control register. Your code must:

1. Assign the value 1 to the data direction bit (bit 6). (This makes the microprocessor pin an output pin.)
2. Assign the value 0 to the mode bit (bit 4). (This configures general-purpose I/O mode.)
3. To turn the LED ON, assign the value 0 to the data bit (bit 7), or
4. To turn the LED OFF, assign the value 1 to the data bit (bit 7).

For example, this line of C code will turn ON LED ST0:

```
*(unsigned short) (0x10005000) = 0x0040;
```

This line of code will turn OFF LED DS3:

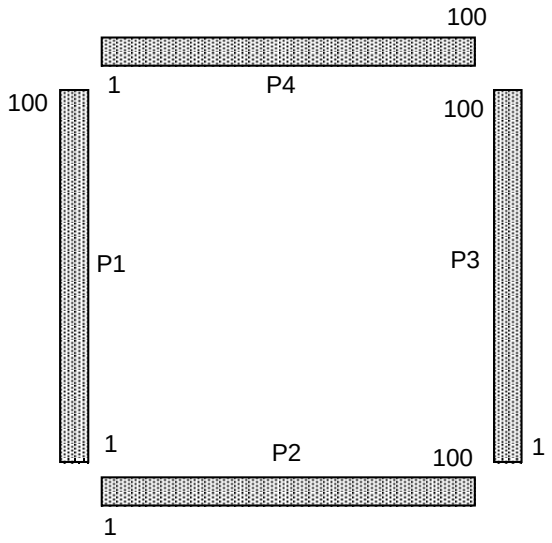
```
*(unsigned short) (0x10005010) = 0x00C0;
```

Section 4 Connector Information

This chapter consists of pin assignments and signal descriptions for CMB1200 MAPI and OnCE connectors. (This chapter does not cover the standard connectors J8, J9, or J10.)

4.1 MAPI Connectors (P1/J1, P2/J2, P3/J3, P4/J4)

Connectors P1 through P4, all 2-by-50-pin connectors, are the CMB1200 MAPI connectors. (Connectors J1 through J4, on the bottom of the CMB1200, have the same pin assignments.) The diagram below shows the orientation of the CMB1200 MAPI connectors. **Figure 4-1** through **Figure 4-4**, and **Table 4-1** through **Table 4-4**, give the pin assignments and signal descriptions for these connectors.



P1/J1

NC	100	•	•	99	NC				
NC	98	•	•	97	NC	NC	50	• •	49	NC
NC	96	•	•	95	NC	GFLCS*	48	• •	47	SRSIZ2*
NC	94	•	•	93	NC	GND	46	• •	45	SRSIZ1*
GND	92	•	•	91	NC	GALB1*	44	• •	43	SRSIZ0*
VCC	90	•	•	89	GND	GALB0*	42	• •	41	GND
NC	88	•	•	87	NC	GND	40	• •	39	NC
NC	86	•	•	85	NC	INT 6	38	• •	37	NC
NC	84	•	•	83	NC	INT 4	36	• •	35	INT 7
NC	82	•	•	81	NC	GND	34	• •	33	INT 5
NC	80	•	•	79	NC	INT 2	32	• •	31	INT 3
NC	78	•	•	77	NC	INT 0	30	• •	29	INT 1
NC	76	•	•	75	NC	GND	28	• •	27	NC
GND	74	•	•	73	NC	GND	26	• •	25	NC
NC	72	•	•	71	GND	NC	24	• •	23	NC
NC	70	•	•	69	NC	NC	22	• •	21	NC
NC	68	•	•	67	NC	NC	20	• •	19	NC
NC	66	•	•	65	NC	NC	18	• •	17	NC
NC	64	•	•	63	NC	NC	16	• •	15	GND
NC	62	•	•	61	NC	NC	14	• •	13	NC
NC	60	•	•	59	NC	GND	12	• •	11	NC
GND	58	•	•	57	NC	GND	10	• •	9	COL 7
NC	56	•	•	55	NC	COL 6	8	• •	7	COL 5
NC	54	•	•	53	NC	COL 4	6	• •	5	COL 3
GND	52	•	•	51	NC	COL 2	4	• •	3	COL 1
						GND	2		1	COL 0

Figure 4-1 MAPI Connector P1/J1 Pin Assignments

Table 4-1 MAPI Connector P1/J1 Signal Descriptions

Pin	Mnemonic	Signal
100 – 93, 91, 88 – 75, 73, 72, 70 – 59, 57 – 53, 51 – 49, 39, 37, 27, 25 – 16, 14, 13, 11	NC	No connection
92, 89, 74, 71, 58, 52, 46, 41, 40, 34, 28, 26, 15, 12, 10, 2	GND	GROUND
90	VCC	OPERATING VOLTAGE — Transmission line for +5-volt CMB1200 input power.
48	GFLCS*	GAL FLASH CHIP SELECT — If low, this signal indicates address-decoded chip selection of FLASH.
47	SRSIZ2*	SRAM SIZE — If low, this signal indicates that the CMB1200 has a 64K-by-16 SRAM.
45	SRSIZ1*	SRAM SIZE — If low, this signal indicates that the CMB1200 has a 128K-by-16 SRAM.
44	GALB1*	GAL BANK CHIP SELECT — If low, this signal indicates address-decoded chip selection of SRAM bank 1.
43	SRSIZ0*	SRAM SIZE — If low, this signal indicates that the CMB1200 has a 256K-by-16 SRAM.
42	GALB0*	GAL BANK CHIP SELECT — If low, this signal indicates address-decoded chip selection of SRAM bank 0.
38, 36, 35, 33 – 29	INT 6, INT 4, INT 7, INT 5, INT 2, INT 3, INT 0, INT 1	EXTERNAL INTERRUPTS (lines 6, 4, 7, 5, 2, 3, 0, 1) — Bidirectional interrupt lines that form the external interface to the general-purpose I/O module.
9 – 3, 1	COL 7, COL 6 – COL 1, COL 0	COLUMN STROBES (lines 7–0) — Keypad column strobe lines, open-drain selectable via software. (Default state upon reset is general-purpose input.)

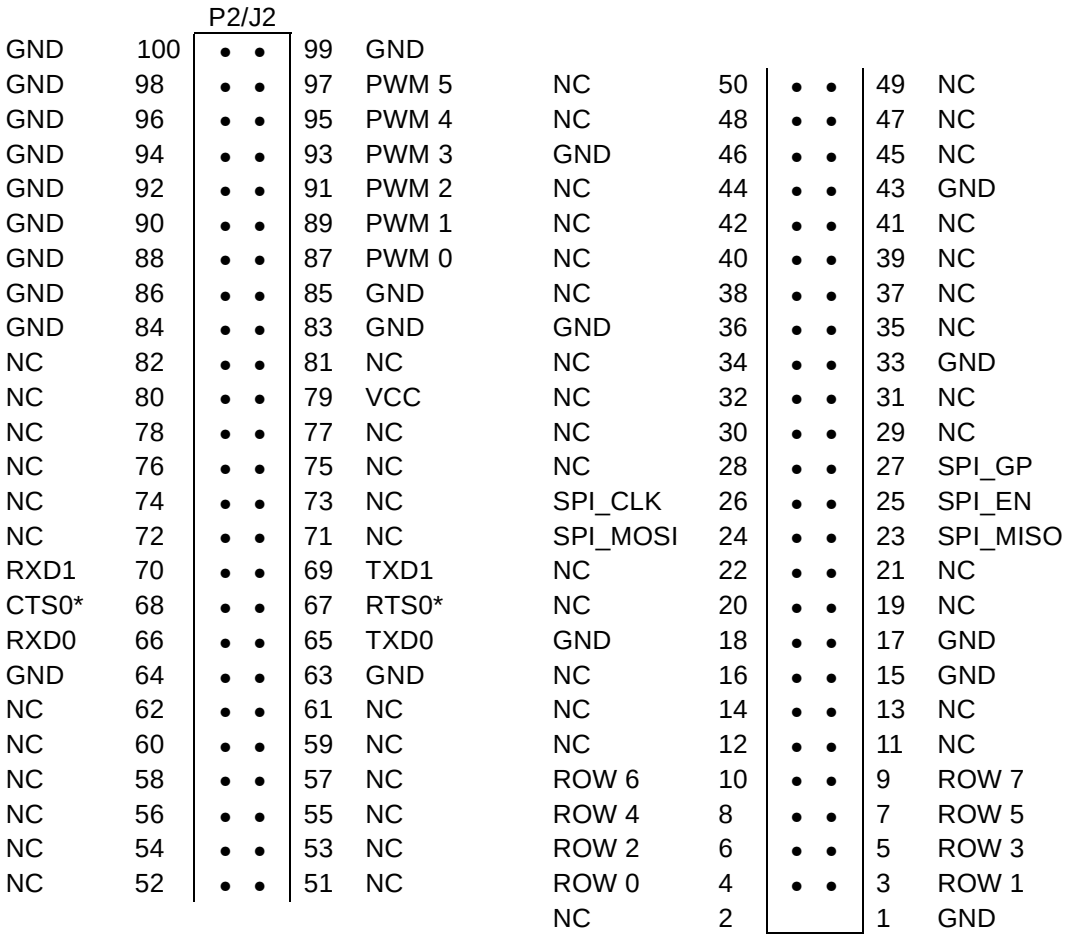

P2/J2
GND 100 99 GND
GND 98 97 PWM 5 NC 50 49 NC
GND 96 95 PWM 4 NC 48 47 NC
GND 94 93 PWM 3 GND 46 45 NC
GND 92 91 PWM 2 NC 44 43 GND
GND 90 89 PWM 1 NC 42 41 NC
GND 88 87 PWM 0 NC 40 39 NC
GND 86 85 GND NC 38 37 NC
GND 84 83 GND GND 36 35 NC
NC 82 81 NC NC 34 33 GND
NC 80 79 VCC NC 32 31 NC
NC 78 77 NC NC 30 29 NC
NC 76 75 NC NC 28 27 SPI_GP
NC 74 73 NC SPI_CLK 26 25 SPI_EN
NC 72 71 NC SPI_MOSI 24 23 SPI_MISO
RXD1 70 69 TXD1 NC 22 21 NC
CTS0* 68 67 RTS0* NC 20 19 NC
RXD0 66 65 TXD0 GND 18 17 GND
GND 64 63 GND NC 16 15 GND
NC 62 61 NC NC 14 13 NC
NC 60 59 NC NC 12 11 NC
NC 58 57 NC ROW 6 10 9 ROW 7
NC 56 55 NC ROW 4 8 7 ROW 5
NC 54 53 NC ROW 2 6 5 ROW 3
NC 52 51 NC ROW 0 4 3 ROW 1
NC 2 1 GND

Figure 4-2 MAPI Connector P2/J2 Pin Assignments

Table 4-2 MAPI Connector P2/J2 Signal Descriptions

Pin	Mnemonic	Signal
100 – 98, 96, 94, 92, 90, 88, 86 – 83, 64, 63, 46, 43, 36, 33, 18, 17, 15, 1	GND	GROUND
97, 95, 93, 91, 89, 87	PWM 5 – PWM 0	PULSE WIDTH MODULATOR (lines 5–0) — External interface lines for the pulse width modulator block. (Default state upon reset is general-purpose input.)
82 – 80, 78 – 71, 62 – 47, 45, 44, 42 – 37, 35, 34, 32 – 28, 22 – 19, 16, 14 – 11, 2	NC	No connection
79	VCC	OPERATING VOLTAGE — Transmission line for +5-volt CMB1200 input power.
70, 66	RXD1, RXD0	RECEIVE DATA (lines 1, 0) — Input data receive lines for UART channels 1 and 0.
69, 65	TXD1, TXD0	TRANSMIT DATA (lines 1, 0) — Output data transmission lines for UART channels 1 and 0.
68	CTS0*	CLEAR TO SEND 0 — Active-low output that can be programmed as the UART-channel-0 clear-to-send signal.
67	RTS0*	REQUEST TO SEND 0 — Active-low input that can be programmed as the UART-channel-0 request-to-send signal.
27	SPI_GP	SPI GENERAL-PURPOSE OUTPUT — Serial peripheral interface module output line: a control line for external logic or devices.
26	SPI_CLK	SPI SERIAL CLOCK — Serial shift clock line for the serial peripheral interface module.
25	SPI_EN	SPI ENABLE — In master mode, the peripheral chip-select line. In slave mode, the slave enable line.
24	SPI_MOSI	SPI DATA MASTER OUT/SLAVE IN — In master mode, serial data output line from the serial peripheral interface module of the MCU. In slave mode, serial data input line to the serial peripheral interface.
23	SPI_MISO	SPI DATA MASTER IN/SLAVE OUT — In master mode, serial data input line to the serial peripheral interface of the MCU. In slave mode, serial data output line from the serial peripheral interface.
10 – 3	ROW 6, ROW 7, ROW 4, ROW 5, ROW 2, ROW 3, ROW 0, ROW 4	ROW SENSES (lines 6, 7, 4, 5, 2, 3, 0, 1) — Keypad row sense lines. (On-chip 47KΩ pull-up resistors are connected to these lines; the default state upon reset is general-purpose input.)

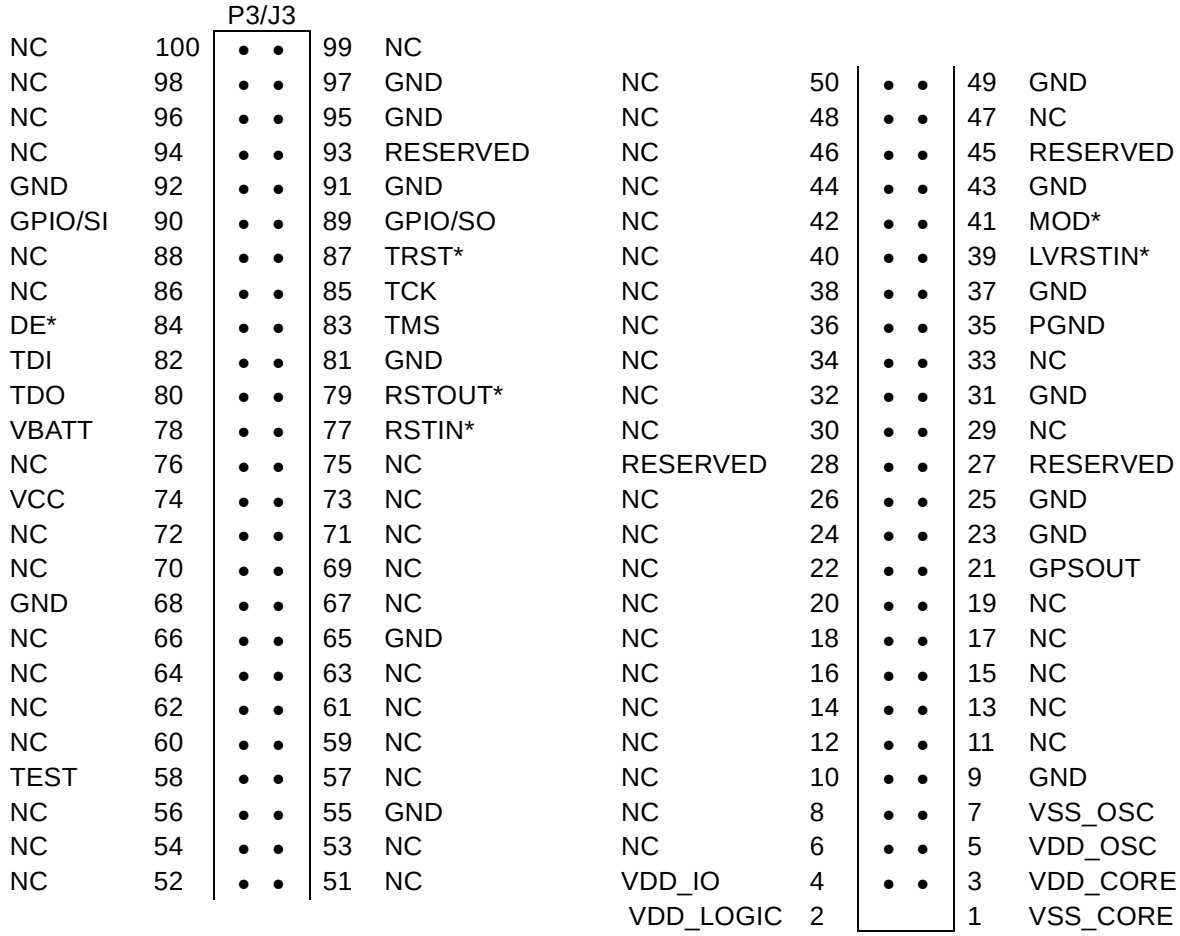


Figure 4-3 MAPI Connector P3/J3 Pin Assignments

Table 4-3 MAPI Connector P3/J3 Signal Descriptions

Pin	Mnemonic	Signal
100 – 98, 96, 94, 88, 86, 76, 75, 73 – 69, 67, 66, 64 – 57, 56, 54 – 50, 48 – 46, 44, 42, 40, 38, 36, 34 – 32, 30, 29, 26, 24, 22, 20 – 10, 8, 6	NC	No connection
97, 95, 92, 91, 81, 68, 65, 55, 49, 43, 37, 35, 31, 25, 23, 9	GND	GROUND
93, 45, 28, 27	RESERVED	RESERVED
90, 89	GPIO/SI, GPIO/SO	GENERAL-PURPOSE I/O SERIAL IN, OUT — General-purpose input and output lines.
87	TRST*	TEST RESET — Active-low input signal to the Schmitt trigger, asynchronously initializing the test controller. The TRST* pin has an internal 47k pullup resistor.
85	TCK	TEST CLOCK — Input signal that synchronizes the JTAG test logic. The TCK pin has an internal 47k pull-up resistor.
84	DE*	DEBUG EVENT — Open-drain, active-low debug signal. If an input signal from an external command controller, causes the EVB to enter debug mode. If an output signal, acknowledges that the MCU is in debug mode.
83	TMS	TEST MODE SELECT — Input signal that sequences the test controller's state machine, sampled on the rising edge of the TCK signal. The TMS pin has an internal 47k pull-up resistor.
82	TDI	TEST DATA INPUT — Serial input signal for test instructions and data, sampled on the rising edge of the TCK signal. The TDI pin has an internal 47k pullup resistor.
80	TDO	TEST DATA OUTPUT — Serial output signal for test instructions and data. Three-stateable and actively driven in the Shift-IR and Shift-DR controller states, this signal changes on the falling edge of the TCK signal.
79	RSTOUT*	RESET OUT — Active-low output signal that resets external components. Activation of any internal reset sources asserts this line.
78	VBATT	STANDBY BATTERY POWER
77	RSTIN*	RESET IN — Active-low input signal that starts a system reset: a reset of the PowerStrike device and most peripherals. This signal does not affect the debug module (which the system provides via the TRST* line).
74	VCC	OPERATING VOLTAGE — Transmission line for +5-volt CMB1200 input power.
58	TEST	FACTORY TEST MODE — Input signal that selects factory test mode. The TEST pin has an internal 100k pulldown resistor.

Table 4-3 MAPI Connector P3/J3 Signal Descriptions (Continued)

Pin	Mnemonic	Signal
41	MOD*	INITIAL VECTOR SELECT — If high, selects the initial reset vector from the lowest address of internal ROM. If low, selects the initial reset vector from the lowest address of chip select 0, then jumps to that address.
39	LVRSTIN*	LOW VOLTAGE RESET IN — Active-low input signal that starts a system reset: a reset of the PowerStrike chip and most peripherals. This signal also selects the backup power supply source for the RAM array and for the OSC/time-of-day timer. This signal does not affect the debug module (which the system provides via the TRST* line).
7	VSS_OSC	OSCILLATOR GROUND — Ground line for the MCU oscillator.
5	VDD_OSC	OSCILLATOR VOLTAGE — Transmission line for +3.3-volt MCU oscillator operating power.
4	VDD_IO	I/O VOLTAGE — Transmission line for +3.3-volt MCU I/O operating power.
3	VDD_CORE	CORE VOLTAGE — Transmission line for +3.3-volt (or alternate +2.0-volt) MCU core operating power.
2	VDD_LOGIC	LOGIC VOLTAGE — Transmission line for +3.3-volt CMB1200 logic operating power.
1	VSS_CORE	CORE GROUND — Ground line for the MCU core.

P4/J4

NC	100	• •	99	NC					
NC	98	• •	97	GND	A8	50	• •	49	A9
GND	96	• •	95	CLKOUT	A6	48	• •	47	A7
NC	94	• •	93	GND	A4	46	• •	45	A5
NC	92	• •	91	CS3	A2	44	• •	43	A3
GND	90	• •	89	CS2*	A0	42	• •	41	A1
OE*	88	• •	87	CS1*	GND	40	• •	39	GND
NC	86	• •	85	CS0*	NC	38	• •	37	NC
NC	84	• •	83	GND	NC	36	• •	35	NC
EB1*	82	• •	81	R/W*	NC	34	• •	33	NC
EB0*	80	• •	79	NC	NC	32	• •	31	NC
NC	78	• •	77	NC	NC	30	• •	29	NC
GND	76	• •	75	GND	GND	28	• •	27	GND
NC	74	• •	73	NC	NC	26	• •	25	NC
NC	72	• •	71	NC	NC	24	• •	23	NC
NC	70	• •	69	NC	NC	22	• •	21	NC
NC	68	• •	67	NC	D14	20	• •	19	D15
NC	66	• •	65	NC	D12	18	• •	17	D13
A20	64	• •	63	A21	GND	16	• •	15	GND
A18	62	• •	61	A19	D10	14	• •	13	D11
A16	60	• •	59	A17	D8	12	• •	11	D9
GND	58	• •	57	GND	D6	10	• •	9	D7
A14	56	• •	55	A15	D4	8	• •	7	D5
A12	54	• •	53	A13	D2	6	• •	5	D3
A10	52	• •	51	A11	D0	4	• •	3	D1
					NC	2		1	NC

Figure 4-4 MAPI Connector P4/J4 Pin Assignments

Table 4-4 MAPI Connector P4/J4 Signal Descriptions

Pin	Mnemonic	Signal
100 – 98, 94, 92, 86, 84, 79 – 77, 74 – 65, 38 – 29, 26 – 21, 2, 1	NC	No connection
97, 96, 93, 90, 83, 76, 75, 58, 57, 40, 39, 28, 27, 16, 15	GND	GROUND
95	CLKOUT	CLOCK OUTPUT — An external clock source: LO_REFCLK or HI_REFCLK.
91, 89, 87, 85	CS3 – CS0*	CHIP SELECTS (lines 3–0) — Output lines that provide chip selects to external devices. Note that CS3 is active high, but the others are active low.
88	OE*	OUTPUT ENABLE — Active-low signal that indicates that a bus access is a read access; enables slave devices to drive the data bus.
82, 80	EB1*, EB0*	ENABLE BYTES 1, 0 — Active-low outputs active during an operation to corresponding data bits (D7–D0 for enable byte 1, D15–D8 for enable byte 0). You can configure these bytes to assert for write cycles or for both read and write cycles.
81	R/W*	READ/WRITE ENABLE — Active-low signal that indicates whether the current bus access is a read access or write access.
64 – 59, 56 – 41	A20, A21, A18, A19, A16, A17, A14, A15, A12, A13, A10, A11, A8, A9, A6, A7, A4, A5, A2, A3, A0, A1	ADDRESS BUS (lines 21–0) — Output lines for addressing external devices. These lines change state only during external-memory accesses.
20 – 17, 14 – 3	D14, D15, D12, D13, D10, D11, D8, D9, D6, D7, D4, D5, D2, D3, D0, D1	DATA BUS (lines 15–0) — Bidirectional data lines for accessing external memory. A hardware reset or no external-bus activity holds these lines in their previous logic state.

4.2 OnCE Connector (P5)

Connector P5, a 2-by-7-pin connector, conveys data and control signals to and from the OnCE control block. **Figure 4-5** and **Table 4-5** give the pin assignments and signal descriptions for this connector.

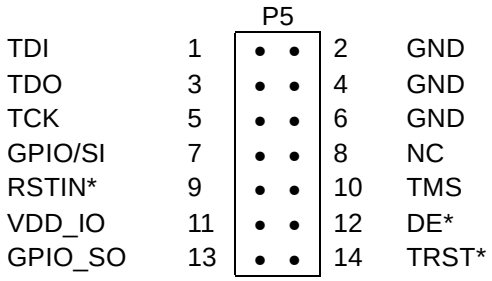


Figure 4-5 OnCE Connector P5 Pin Assignments

Table 4-5 OnCE Connector P5 Signal Descriptions

Pin	Mnemonic	Signal
1	TDI	DEBUG SERIAL INPUT — Data and command input line to the OnCE controller.
2, 4, 6	GND	GROUND
3	TDO	DEBUG SERIAL OUTPUT — Serial data output line from the OnCE controller.
5	TCK	DEBUG SERIAL CLOCK — Serial clock input line to the OnCE control block.
7	GPIO/SI	GENERAL PURPOSE I/O SERIAL IN — General purpose input line to the OnCE controller.
8	NC	No connection
9	RSTIN*	RESET IN — Active-low input line to the OnCE controller, signalling a reset.
10	TMS	DEBUG MODE SELECT — Input signal that tells the OnCE control block to advance one mode state (of the cycle of mode states).
11	VDD_IO	OPERATING VOLTAGE — Transmission line for +3.3-volt MCU operating power.
12	DE*	DEBUG EVENT — Active-low debug-mode control line for the OnCE controller. An input signal from an external command controller makes the OnCE controller immediately enter debug mode. An output signal acknowledges debug-mode-entry to the external command controller.
13	GPIO_SO	GENERAL PURPOSE I/O SERIAL OUT — General purpose output line from the OnCE controller.
14	TRST*	TEST RESET — Active-low input line for an external reset signal to the OnCE controller.

Connector Information

C

CMB1200
- foot placement 12
- layout 10, 11

computer system connections 18, 19
configuration 13–20
connections, computer system 18, 19
connector pin assignments
- MAPI connectors P1/J1-P4/J4 34, 36, 38, 41
- OnCE connector P5 43

connector signal descriptions
- MAPI connectors P1/J1-P4/J4 35, 37, 39, 42
- OnCE connector P5 43

controlling LEDs 31, 32
core voltage jumper header (W19) 18

D

debugging embedded code 21–25

F

features 9, 10
FLASH byte/word select jumper headers (W3, W6) 16
FLASH programmer
- restoring system software 31
- steps 26–30

foot placement 12

I

introduction 9–12

J

jumper headers
- positions 13–15
- setting 13–18

L

layout, CMB1200 11
LEDs, controlling 32

M

memory maps 20

O

operation 21–32

P

Picobug debug monitor
- commands 22
- sample session 23–25
- using 21–23

pin assignments
- MAPI connectors P1/J1-P4/J4 34, 36, 38, 41
- OnCE connector P5 43

positions, jumper headers 13–15

R

requirements, system/user 10
restoring system software 31

S

self-test, performing 19
serial port enable jumper header (W14) 17
setting jumper headers
- core voltage (W19) 18
- FLASH byte/word select (W3, W6) 16
- positions 13–15
- serial port enable (W14) 17
- SRAM/FLASH chip select (W1) 15
- startup vector select (W9) 16

signal descriptions
- MAPI connectors P1/J1-P4/J4 35, 37, 39, 42
- OnCE connector P5 43

specifications 11
SRAM/FLASH chip select jumper header (W1) 15
startup vector jumper header (W9) 16
system requirements 10
system software, restoring 31

U

user requirements 10

Index

Revision History

Revision Number	Date	Author	Summary of Changes
Original	May 1998	MTC DDOC	Original document.
1	Dec 1998	MTC DDOC	P. ii — New ESD protection caution. Pp. 3-7—3-14 — Explanation of the new FLASH programmer replaced the original manual's explanation of earlier software.
2	May 1999	MTC DDOC	P. 3 — Mention of the 5-volt power supply. Pp. 20—25 — Updates to FLASH programmer information, including restoring system software.

Revision History

This manual is a product of the Motorola M•CORE Technology Center Design Documentation team. Technical writing, illustration, and production editing performed with Adobe® Framemaker® running on multiple platforms. Printed by Ken Cook, Inc. in Milwaukee, Wisconsin.